Approximating Action-Value Functions: Addressing Issues of Dynamic Range

Mance E. Harmon
Adaptive Network Laboratory
Department of Computer Science
University of Massachusetts
December 17, 1998

ABSTRACT

Function approximation is necessary when applying RL to either Markov decision processes (MDPs) or semi-Markov decision processes (SMDPs) with very large state spaces. An often overlooked issue in approximating Q-functions in either framework arises when an action-value update in a given state causes a large policy change in other states. Another way of stating this is to say that a small change in the Q-function results in a large change in the implied greedy policy. We call this sensitivity to changes in the Q-function the *dynamic range problem* and suggest that it may result in greatly increasing the number of training updates required to accurately approximate the optimal policy. We demonstrate that Advantage Learning solves the dynamic range problem in both frameworks, and is more robust than some other RL algorithms on these problems. For an MDP, the Advantage Learning algorithm addresses this issue by re-scaling the dynamic range of action values within each state by a constant. For SMDPs the scaling constant can vary for each action.

1. INTRODUCTION

Reinforcement learning (RL) systems are commonly used to solve Markov Decision Processes (MDPs), tasks in which the interval between decisions is fixed. RL systems should be general enough to solve tasks that require performing actions at a mixture of time scales (i.e., the RL system makes decisions concerning low-level, as well as high-level, actions). Such tasks are commonly called semi-Markov decision processes (SMDPs) and differ from MDPs in that the interval between decisions varies. People sometimes use hierarchies for solving large MDPs. When doing so, the problem is by definition an SMDP.

One way to solve MDPs and SMDPs is to find an approximation of the action-value function for the task. The optimal policy can easily be extracted by choosing the action with the greatest approximated value in each state. One of the most common RL algorithms for finding action-value functions is Q-learning (Watkins, 1989). For large MDPs and SMDPs it may be necessary to combine Q-learning with general function approximation to find approximations of the action-value functions (Q-functions).

For a given function approximator and task (MDP or SMDP), it may be the case that an action-value update in a given state always causes a large policy change in other states. In other words, a small change in the Q-function will result in a large change in the implied greedy policy. We call this the *dynamic range problem* and suggest that it may result in greatly increasing the number of training updates required to accurately approximate the optimal policy.

We demonstrate that *Advantage Learning* solves the dynamic range problem in both MDP and SMDP frameworks, and is more robust than Q-learning on these problems. *Advantage Learning* and its forerunner, *Advantage Updating*, a generalization of the *Q-learning* algorithm, have been discussed previously in the context of continuous-time MDPs (Puterman, 1994) in Baird (1993, 1994), Harmon, Baird, and Klopf (1995, 1996), Baird, Harmon, and Klopf (1996), and Harmon and Baird (1996a, 1996b).[1] Here we present a general framework for discussing issues of dynamic range in action-value functions. We then use this framework for analyzing the action-value functions learned by the Q-learning algorithm for both MDPs and SMDPs, which we call Q-functions following common practice. We describe features we consider desirable in action-value functions and derive an operator that maps a given Q-function to our desired function, the *Advantage* function. We provide theoretical justification and empirical evidence of the desirability of approximating Advantage functions instead of Q-functions.

Section 2 provides the notation used throughout this paper and background information on MDPs and RL. Section 3 discusses issues involved in accurately approximating Q-functions. In Section 4 the concept of an Advantage function is derived and empirical results are presented for both MDPs and SMDPs that illustrate the properties of this function. Section 5 presents alternatives to using Advantage functions and includes closing remarks.

2. BACKGROUND AND NOTATION

RL systems typically use a set of real-valued parameters to store the information that is learned. When a parameter is updated during learning, the notation $w \leftarrow k$ represents the operation of instantaneously changing the parameter w so that its new value is k, whereas $w \xleftarrow{\alpha} k$ represents the operation of moving the value of w toward k. This is equivalent to $w_{new} \leftarrow (1-\alpha)w_{old} + \alpha k$ where the step size parameter α is a small positive number.

The functions stored in a learning system at a given time are represented by variables without superscripts such as π, V, A, or Q. The optimal functions that are being approximated are represented by * superscripts, such as π^*, V^*, A^*, or Q^*.

[1] A summary of these results is given in Appendix A.

2.1 Markov Decision Processes

A *Markov decision process* (MDP) is a system that changes its state based upon its current state and an action chosen by a controller. The set of possible states and the set of possible actions may each be finite or infinite. At time *t*, the controller chooses an action, u_t, based upon the state of the MDP, x_t. The MDP then transitions to a new state, x_{t+1}. The state transition may be stochastic, but the probability, $P(u_t, x_t, x_{t+1})$, of transitioning from state x_t to state x_{t+1} after receiving action u_t is a function of only x_t, x_{t+1}, and u_t, and is not affected by previous states or actions. The MDP also sends the controller a scalar value known as a reward. The expected reward received by the controller as a result of the transition from state x_t to x_{t+1} is $R(x_t, u_t)$.

A *policy*, π, is a function that specifies a particular action for the controller to perform in each state *x*. The *expected total discounted reward* associated with a state *x* is tne expected sum of rewards received when starting in that state: $E\left\{\sum_{t=0}^{\infty} \gamma^t r_t | x_0 = x\right\}$, where r_t is the reward received at time step *t*. The discount factor γ, $0 \le \gamma \le 1$, is a parameter that determines the relative significance of earlier versus later reward. The *value* of a state *x*, $V^\pi(x)$, is the expected total discounted reward received for starting in state *x* and choosing all actions according to a given policy π. The value of an action, $Q^\pi(x,u)$, is the expected total discounted reward for starting in state *x*, performing action *u*, and then choosing all actions according to the given policy π.

An *optimal policy*, π^*, for a given MDP is a policy such that choosing $u_t = \pi^*(x_t)$ results in maximizing the expected total discounted reward for any choice of starting state. If there are a finite number of states and actions, and $\gamma < 1$, then at least one optimal policy is guaranteed to exist.

2.2 State-Value Functions and Action-Value Functions

Roughly, the goal of RL is to find an optimal policy, π^*. Policies can be extracted from either state-value functions (functions of state only) or action-value functions (functions of both state and action). State-value functions are functions of only state; action-value functions are functions of both state and action. Consequently, RL algorithms can generally be grouped into two categories: those that approximate state-value functions, and those that approximate action-value functions.

The goal of the RL system is to find a function V that satisfies the following equation for all x_t:

$$V(x_t) = \max_u E\left[R(x_t, u) + \gamma V(x_{t+1})\right], \tag{1}$$

where E indicates the expected value of performing action u in state x_t.

The unique solution to equation 1 is the optimal state-value function V^*. Equation 1 is called the Bellman equation in dynamic programming (Bertsekas, 1987). The optimal policy, π^*, can be extracted from V^* by letting $\pi(x_t) = \arg\max_u E\left[R(x_t, u) + \gamma V^*(x_{t+1})\right]$.

The goal of an RL system using Q-learning is to find a function Q that satisfies the following equation for all (x,u) pairs:

$$Q(x_t, u_t) = E\left[R(x_t, u_t) + \gamma \max_{u_{t+1}} Q(x_{t+1}, u_{t+1})\right]. \tag{2}$$

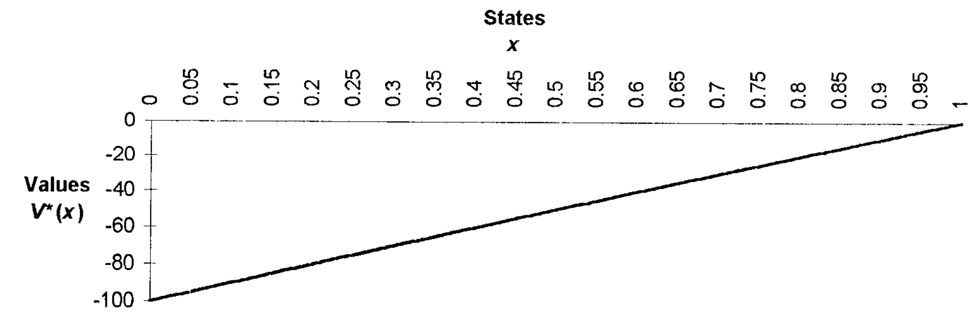
The unique solution to Equation 2 is the optimal action-value function, Q^*. Equation 2 is the Bellman equation for Q-learning. The optimal policy, π^*, can be extracted from Q^* by letting $\pi(x_t) = \arg\max_u Q^*(x_t, u)$.

3. FOUNDATIONS: POLICY REPRESENTATION, DYNAMIC RANGE, AND SENSITIVITY

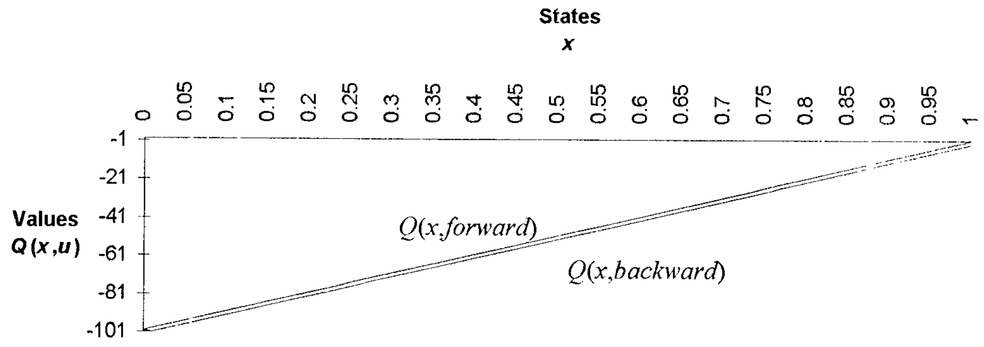
3.1. The Method of Extracting π Has Significant Consequences

The choice of function to approximate V^* or Q^* can have a profound effect on the degree of accuracy needed in the approximation before the implied policy, π, equals the optimal policy, π^*, in all states. To explain further and develop intuition, we present the following deterministic MDP. The set of states is $X = \{0,.01,.02,\ldots 1\}$. There is a single absorbing state, x=1.

This MDP can be visualized as a hallway with a door at one end (x=1). Given any initial state, the goal is to maximize the total undiscounted reward necessary to exit the hallway. Two actions are possible in every state: step forward ($x_{t+1} = x_t + .01$), or step backward ($x_{t+1} = x_t - .01$), with the exception of the state x=0 in which both actions result in x=0.01. Each action results in a negative unit reward (-1). The optimal state-value function, V^*, and optimal Q-function, Q^*, for this MDP are graphed below.



State-Value Function



Q-function

Assuming that the state-value function is approximated by an affine function of the state, a very small degree of accuracy is needed in the state-value function approximation before $\pi(x) = \pi^*(x)$ for all x. All that is required is for the slope of the approximation to have the correct sign. In other words, if $V(x+1) > V(x)$ for all x, then $\pi(x) = \pi^*(x)$ for all x. Because we have assumed a function approximator that generalizes well, very few training samples will be needed to achieve this result.

However, this is not the case for the Q- function. Here the policy is extracted by comparing the relative action values within a given state. The action with the greatest value is the policy's choice for that state. This necessarily means that the action the approximation implies has the greatest value must correspond to the action with the greatest value in the optimal Q-function for the same state, and this must be true for all states. In the above example, $Q^*(x,forward) > Q^*(x,backward)$ for all x (with the exception of x=0). Therefore, the approximation, Q, must accurately reflect this ordering for all x. In short, in the state-value function it takes a large change in parameters to change the slope enough to change the policy, but in the Q-function it takes only a tiny change in the parameters to raise one line above the other, resulting in a change in policy.

Generalization is the term often used to refer to the change in the value of one state as a result of training in another state. In the above example, when approximating V^*, generalization helps to quickly find an approximation that results in the correct policy in every state. However, as we demonstrate below, when approximating Q^*, generalization may be a hindrance. Moreover, we demonstrate that one can alleviate the problem with a simple transformation of the Q-function that results in a new action-value function that is far easier to approximate.

3.2. Dynamic Range and Sensitivity

What properties of action-value functions make it difficult to achieve an implied policy of π^*? Should we simply use a different function approximator? Or, for a given function approximator, can we transform the action-value function in such a way that we can achieve an implied policy of π^* with fewer training examples?

To answer these questions, we now introduce a framework for discussing several properties of action-value functions.

3.2.1. Dynamic Range

An important consideration is the relationship between the dynamic range of action values in a given state and the dynamic range of state values. We define two variables, D_{sa} and D_{aa}, where *sa* refers to state-action and *aa* refers to action-action, to represent the two distinct dynamic range ratios. Assuming a finite state set, the dynamic range over state values is defined as

$$d_{sv} = \max_x V(x) - \min_x V(x).$$

Similarly, the dynamic range over all action values within a given state is defined as

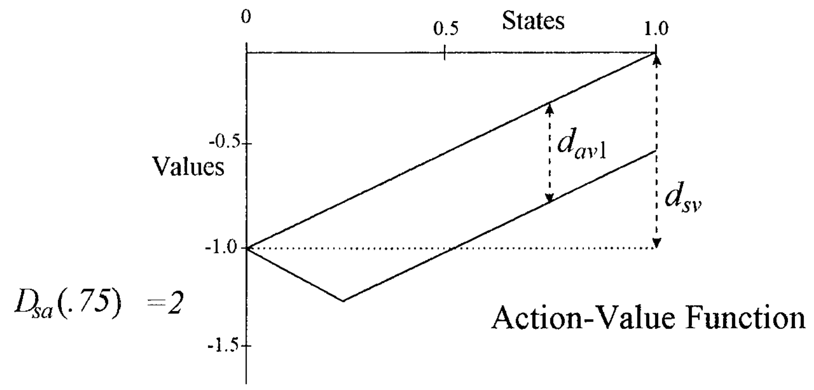
$$d_{av1}(x) = \max_u Q(x,u) - \min_u Q(x,u).$$

We use the subscript of $av1$ for this quantity to distinguish it from a different dynamic range of action values within a given state. Namely, we define $d_{av2}(x,u)$ as

$$d_{av2}(x,u) = \left[\max_u Q(x,u)\right] - Q(x,u).$$

We define the state-action dynamic range ratio, $D_{sa}(x)$, to be $d_{sv}/d_{av1}(x)$ with $D_{sa}(x) = 1$ if $d_{av1}(x) = 0$. The action-action dynamic range ratio, $D_{aa}(x,u)$, is defined as $d_{av1}(x)/d_{av2}(x,u)$, again with $D_{aa}(x,u) = 1$ if $d_{av2}(x,u) = 0$. Although presented here for completeness, our discussion of action-value functions will not include issues of action-action dynamic range ratio, $D_{aa}(x,u)$, until we address functions relevant to the SMDP framework in Section 4.3.

Consider the action-value function presented below. The state-value dynamic range, d_{sv}, is $V(1)-V(0)=0-(-1)=1$. The action value dynamic range over all actions in state .75, $d_{av1}(.75)$, is $\max_u Q(.75,u)-\min_u Q(.75,u)=-.25-(-.75)=.5$. Therefore, $D_{sa}(.75)=1/.5=2$.



3.2.2 Sensitivity

Another property of action-value functions to consider is the degree to which the implied policy is changed as a result of a training update of the function approximator. The degree to which the policy changes is a function of both the function approximator and Q^*. Here, we want to know how a different choice of Q^* would affect this policy sensitivity.

Let $g(x)$ represent the difference between the value of the state, $V(x)$, and the maximum action value over the sub-optimal actions:

$$g(x)=V(x)-\max_{u|u\in U,Q(x,u)\neq V(x)} Q(x,u).$$

Given $g(x)$ we can define $\Delta g(x)$, the fractional change in $g(x)$ resulting from a single update to the parameter vector of the function approximator transforming $g(x)$ to $g(x')$:

$$\Delta g_{t,t+1}(x)=\frac{g(x')-g(x)}{g(x)}.$$

Define the error in the approximation of the action value to be $E(x,u)=|Q^*(x,u)-Q(x,u)|$, the fractional change in the error as a result of a single update to the parameter vector of the function approximator upon training on the state-action pair (x,u) is

$$\Delta E(x,u)=\frac{E'(x,u)-E(x,u)}{E(x,u)}.$$

For any triple (x',x,u), $x,x'\in X$, $x\neq x'$, $u\in U$, we define *sensitivity*, $S(x',x,u)$, to be the amount that g changes in state x' as a result of the change in the error in the approximation of an action value in a different state x:

$$S(x',x,u)=\frac{\Delta g(x')}{\Delta E(x,u)}.$$

If sensitivity is small, then the change in the error in the approximation of the value for state-action pair (x,u) will have little effect on g in state x'. If sensitivity is high, then a small change in the approximation of the value for state-action pair (x,u) will have a large effect on g in state x'. If $S(x',x,u)>1$, then training enough in (x,u) to eliminate the error there changes the policy in state x'. If $S(x',x,u)<1$, then training enough in (x,u) to eliminate the error will not change the policy in state x'.

3.3. Are Sensitivity and Dynamic Range Related?

Consider an MDP with two actions available in each state, u_1 and u_2. The optimal action values $Q^*(x,u_1)$ and $Q^*(x,u_2)$ represent the long-term reward expected when starting in state x and performing action u_1 or u_2 respectively for a single step, followed by optimal actions thereafter. In a typical RL problem with a large (or continuous) state space, it is frequently the case that performing one wrong action in a long sequence of optimal actions has little effect on the total reward. In such a case, $Q^*(x,u_1)$ and $Q^*(x,u_2)$ are relatively close (i.e., a small dynamic range $d_{av1}(x)$). On the other hand, the values of widely separated states typically will not be close to each other. Therefore $\max_u Q^*(x_1,u)$ and $\max_u Q^*(x_2,u)$ may differ greatly for some choices of x_1 and x_2 resulting in a large d_{sv}, which in turn results in a large D_{sa}.

The policy implied by a Q-function for a given state is determined by the relative action values within that state. Consider the case where updating $Q(x,u)$ causes a change in action values in x', and updating $Q(x'',u)$ causes a change in action values in x as we approach the desired function Q^*. Our concern is only with what happens as the approximation Q approaches the desired function Q^*. Early in the learning process it is reasonable to assume large policy changes. If the Q-function is stored in a function approximation system that generalizes, the sensitivity, S, in the implied policy will grow as D_{sa} grows. In cases where the penalty is small for one wrong action in a sequence of many actions, the dynamic range of action values within a given state is small, and the implied policy will be sensitive to generalization.

The change in the difference between the value of the action considered optimal in x' and the maximum value over sub-optimal actions in x' as a result of updating the value of $Q(x,u)$ will likely be large, thus corrupting the learned policy in x'. The policy in state x may accurately reflect the optimal policy after the update, $\pi(x) = \pi^*(x)$. However, because of the large sensitivity, $S(x, x'', u)$, the policy in state x will also likely be corrupted as a result of updating the value for some state-action pair $Q(x'',u)$. Thus, the number of training updates required to achieve an implied policy that is an adequate approximation of π^* may be very large. This problem is not a property of any particular function approximation system; rather, it is inherent in the definition of Q-functions.

3.4 Experiment Set #1

3.4.1. The Hall Problem

For the purpose of illustrating the effects of a large D_{sa}, we consider the following class of finite, deterministic MDPs. Each MDP has $10(2^i)$ states, where i indicates the ith MDP. The set of states, X_i, for MDP M_i is $X_i = \left\{ \frac{j}{5(2^i)} \right\}_{j=0}^{5(2^i)}$. The set of actions possible in each state for every MDP in the class is $\{backward, forward\}$. The one-step dynamics for each MDP are defined by the following:

$$f_i(x,u) = \begin{cases} x - \dfrac{1}{5(2^i)} & \text{if } u = backward \text{ and } x > 0 \\ 0 & \text{if } u = backward \text{ and } x = 0 \\ 1 & \text{if } x = 1 \\ x + \dfrac{1}{5(2^i)} & \text{otherwise} \end{cases}$$

The reward function, R, is defined as $R_i(x,u) = \dfrac{-1}{5(2^i)}$ for all $x \in X_i, u \in \{backward, foward\}$. Finally, for each MDP, there exists a single terminal state, x=1.

Each MDP can be visualized as a hallway with a door at one end (x=1). There are two actions available to the RL system in each state: step forward, and step backward (with the exception of the initial state, in which case both action result in a step forward). The objective is to successfully exit the hallway through the door.

Each MDP differs only in the size of the "step" in the hallway (i.e. the transition distance from x to x'). For example, MDP M_0 has a total of five non-terminal states: X_1={0, 0.2, 0.4, 0.6, 0.8, 1}, with a transition distance Δx of 0.2. MDP M_1 has a total of 10 non-terminal states: X_2={0, 0.1, 0.2, ..., 0.8, 0.9, 1}, with a transition distance of 0.1. The reward for each transition is the negative of the transition distance. Therefore, for M_2, $R_2(x,u) = -0.1$.

Each graph below depicts the optimal action-value function, Q^*, for selected MDPs from this class, given that rewards are discounted by $\gamma^{\Delta x}$. In each experiment the RL system is initially placed at the end of the hallway opposite the door (x=0). Note that for all MDPs the Q-values in state 0 are identical because both actions cause a transition to state $0+\Delta x$.

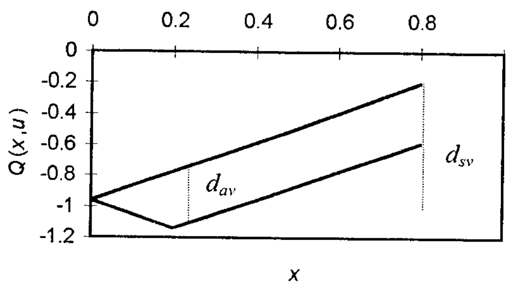


Figure 1: MDP M_0, $D_{sa}\approx 2$ The short dashed line represents d_{av}, the difference between the value of moving forward and backward The long dashed line represents the difference in the maximum and minimum state values over the entire domain.

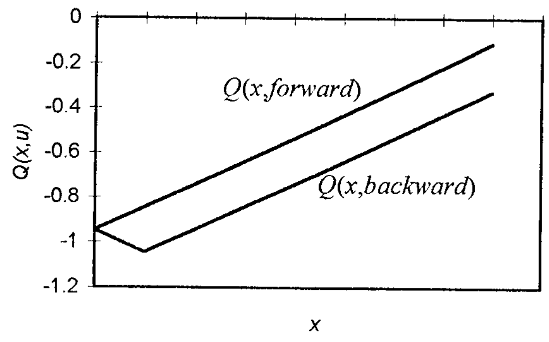


Figure 2: MDP M_1, Dsa≈4 There are twice the number of states in M_1 than in M_0. The value of d_{sv} remained unchanged while the value of d_{av} was reduced by approximately one half in all states other than x=0. Notice this pattern holds in true in Figures 3 and 4 below.

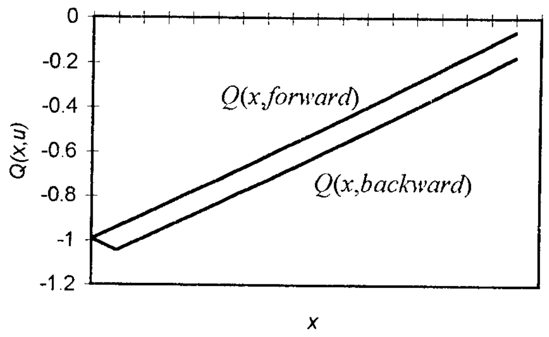


Figure 3: MDP M_2, Dsa≈8

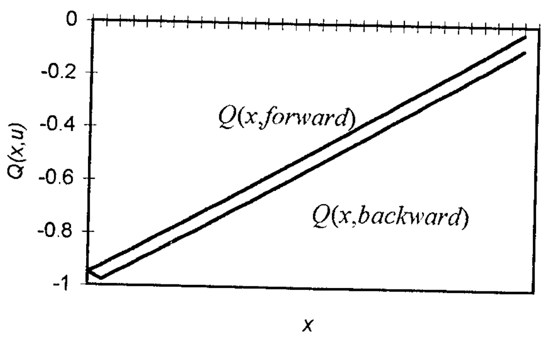


Figure 4: MDP M_3, Dsa≈16

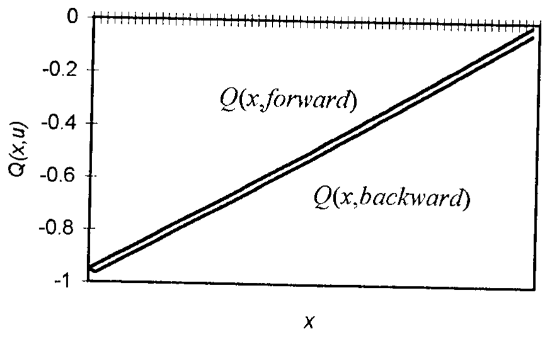


Figure 5: MDP M_4, D_{sa}≈32

Figures 1-5 clearly demonstrate that as the length of the trajectory through state space (measured in state transitions) increases, the significance, with respect to total reward received, of performing a single sub-optimal action decreases. This results in an increase in D_{sa} (i.e., a decrease in d_{av1} for each state relative to the dynamic range for state values, d_{sv}). As stated earlier, the policy for a given state is extracted from the approximation by taking the argmax over action values in that state. Therefore, the action values in each state must be approximated with a degree of accuracy that ensures correct relative values. As D_{sa} increases, the degree of accuracy required in the approximation increases. Equivalently, as D_{sa} increases, the degree of sensitivity, S, in the implied policy increases. When using a function approximator that generalizes, it may become very difficult or impossible to achieve an adequate approximation of π^*.

3.4.2 Empirical Results

We performed sets of supervised learning experiments to demonstrate the validity of this assertion. We chose to use supervised learning rather than RL to clearly demonstrate that the objective action-value functions are at issue, not other characteristics of the MDP such as the number of states. In our experiments the optimal action-value functions of MDPs M_0, M_1, M_2, and M_3 were used as the objective functions. In each experiment, two CMACs (Albus, 1981) were used to represent the action-value function (one for each action). Each CMAC had 10 tilings. Each tiling had 11 tiles, each of which covered 10% of state space (excluding boundaries). In other words, each tile had an effect on 10% of the action values; i.e., generalization extended over a tenth of the domain for all MDPs.

By a *trial* we mean the process of performing parameter updates until the stopping criterion is met. The stopping criterion is to achieve an approximation of the action-value function that implies the optimal policy in every state. Batch weight updates were performed; the parameters of the function approximator were updated only after the presentation of all input-output pairs. The performance measure was the number of weight updates required before achieving the stopping criterion.

For each MDP, the learning rate was optimized. Specifically, for a given learning rate, 10 independent trials were performed (each initialized with a different random number seed), and the average number of updates required to achieve the stopping criterion was determined. The learning rates were then optimized to 2 significant figures. The results are given in Figure 6:

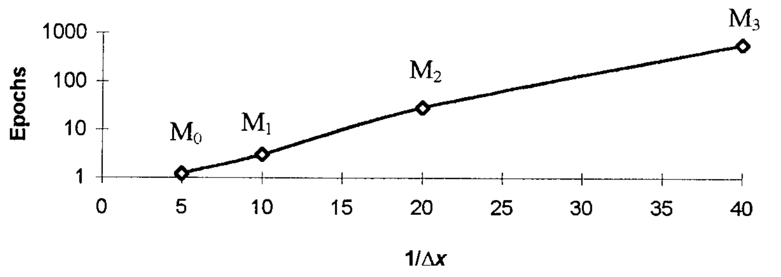


Figure 6: Time To Learn Policy

The number of updates required increases with a decrease in transition distance (i.e., an increase in D_{sa}). Figure 6 shows that the rate of growth in the number of weight updates (as a function of transition distance) required to achieve the stopping criterion is quite large.

4. ADVANTAGE FUNCTIONS

As stated in the previous section, the greedy action implied by an action-value function in a given state is determined by the relative action values within that state. If the function is represented by an approximation system that generalizes across states, the implied policy may be sensitive to the generalization. The degree of sensitivity scales with D_{sa}. In cases where the penalty for one wrong action in a sequence of many actions is small, the dynamic range of action values within a given state, d_{av1}, decreases, and the implied policy becomes even more sensitive to generalization.

One solution to the state-action dynamic range problem would simply be to exaggerate the differences in action values within a state (i.e., re-scale the action-value dynamic ranges, d_{av1} and d_{av2}), while maintaining the state values for all states (i.e., the state-value dynamic range, d_{sv}, remains unchanged), thereby decreasing D_{sa} and decreasing the sensitivity of the implied policy to generalization. Note that simply scaling the function by a constant does re-scale d_{av1}, but it also re-scales d_{sv} by the same amount and therefore does not achieve the goal of decreasing D_{sa}.

4.1 Derivation Of Advantage Function

We begin by defining an operator, F, on the space of action-value functions that achieves our desired goal:

$$F{:}\Theta \to \Theta\text{, where } \Theta = \{f \mid f{:}\, X \times U \to \Re\}.$$

Given a function $c \in \Theta$ such that $c(x,u)>1$ for all $x \in X$ and all $u \in U$, our objective is to find an F such that for all $x \in X$, $u,u' \in U$, and $Q \in \Theta$, two properties are satisfied. Letting $A=F(Q)$, they are:

i) $\max\limits_u A(x,u) = \max\limits_u Q(x,u)$,

ii) $A(x,u') - A(x,u) = c(x,u')\left[Q(x,u') - Q(x,u)\right]$.

Property (*i*) ensures that the state-value dynamic range, d_{sv}, does not change as a result of the transformation. Property (*ii*) ensures that the action-value dynamic range, d_{av}, is increased by a factor of $c(x,u)$, thereby decreasing D_{sa}.

Is there an F that satisfies (*i*) and (*ii*) and, if so, is it unique? We will show that such an F does exist and is unique and is given by:

$$A(x,u) = \max_{u'} Q(x,u') - c(x,u)\left[\max_{u'} Q(x,u') - Q(x,u)\right] \tag{3}$$

where u and u' are arbitrary actions in state x. [2] The derivation of Equation 3 follows:

Rewriting (*ii*), we see that $A(x,u) = c(x,u')Q(x,u) + A(x,u') - c(x,u')Q(x,u')$. Substituting for $A(x,u)$ in (*i*) produces

[2] On first observation it might appear appropriate to rewrite (1) as a weighted average, weighted by c and by $(1-c)$. However, we choose $c>1$ for all state-action pairs.

$$\max_u\left[c(x,u')Q(x,u) + A(x,u') - c(x,u')Q(x,u')\right] = \max_u Q(x,u) \quad \text{for any } u'$$

$$A(x,u') + c(x,u')\left[\max_u Q(x,u) - Q(x,u')\right] = \max_u Q(x,u)$$

$$A(x,u') = \max_u Q(x,u) - c(x,u')\left[\max_u Q(x,u) - Q(x,u')\right]$$

which is the same as $A(x,u) = \max_{u'} Q(x,u') - c(x,u)\left[\max_{u'} Q(x,u') - Q(x,u)\right]$

Thus, for a given function *c* such that $c(x,u)>1$ for all (x,u) pairs, the derivation shows that *F* exists and is the unique operator that has properties (*i*) and (*ii*). We call $A=F(Q)$ the Advantage function (Baird, 1993) derived from *Q*. Note that the *Q*-function is a special case of an Advantage function. If $c(x,u)=1$ for all (x,u), Equation 3 reduces to the original *Q*-function.

Figure 7 graphically demonstrates the results of transforming the *Q*-function into the corresponding Advantage function when *c* is a constant.

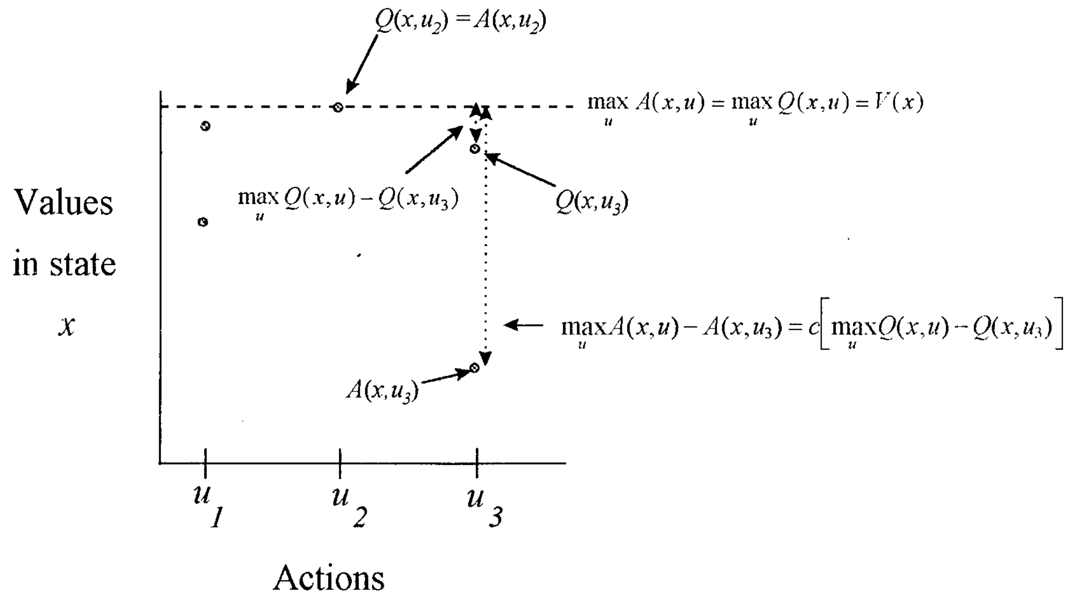

$Q(x,u_2) = A(x,u_2)$
$\max_u A(x,u) = \max_u Q(x,u) = V(x)$
$\max_u Q(x,u) - Q(x,u_3)$
$Q(x,u_3)$
Values in state x
$\max_u A(x,u) - A(x,u_3) = c\left[\max_u Q(x,u) - Q(x,u_3)\right]$
$A(x,u_3)$
u_1
u_2
u_3
Actions

Figure 7: The action values in state *x* before and after transforming the *Q*-function into an Advantage function.

For a single state, *x*, both the original action-value function and the resulting Advantage function are plotted in Figure 7. The dashed line indicates the maximum action-value in state x, which is by definition the value of the state, $V(x)$. As is required by property (i), $\max_u A(x,u) = \max_u Q(x,u)$, which ensures that the policy remains unchanged as a result of the transformation. Neither has d_{sv} been affected by the transformation. Observe that the difference in action values in the original function (Q-function) is small. However, after the transformation, the differences in action values in the new function (Advantage function) have been greatly exaggerated. The dynamic range of action values, d_{av}, within a given state are controlled by c. In other words, d_{av1} has been scaled by a factor of c and d_{sv} has not been changed, so D_{sa} has been reduced by a factor of c, resulting in a diminished degree of sensitivity.

Choosing c

Ideally, one would like to choose the function c such that D_{sa}=1 for all states. A function that satisfies this criterion is called c^*. It will rarely be the case that we have enough information *a priori* to determine c^*. However, it is not difficult to develop heuristic approximations of c^* that result in "good" Advantage functions. For example, if the MDP is an approximation of an underlying continuous time system, then a good approximation of c^* might be $\frac{1}{K\Delta t}$ where Δt is the time step duration for the chosen action and K is an arbitrary constant. As Δt is halved, the change in the total discounted reward received as a result of performing a single sub-optimal action is halved. In other words, d_{av} is halved and D_{sa} is doubled, resulting in increased sensitivity. Choosing $c(x,u) = \frac{1}{K\Delta t}$ counteracts the increase in D_{sa} and causes the underlying Advantage function to remain independent of Δt for small Δt.

4.2 Example: Hall Problem

To illustrate that the policy implied by the Advantage function is less sensitive to generalization than the original Q-function, we return to the Hall Problem described in Section 3.4. Here we describe a set of experiments identical to those presented in Section 3.4.2. with two exceptions: the function approximated was the Advantage function and no optimization was performed on the learning rate. The optimal scaling function c^* was approximated by letting $c(x,u)$ equal $K/\Delta x$ for all (x,u) pairs, where Δx is the transition distance and K=0.2 is a scaling factor chosen to cause Q^* and A^* to be the same function for MDP M_0. For example, $c(x,u)=K/\Delta x=.2/.2=1$ for all (x,u) pairs in M_0, which results in $A(x,u)=Q(x,u)$ for all state-action pairs. The results are summarized in Figure 8.

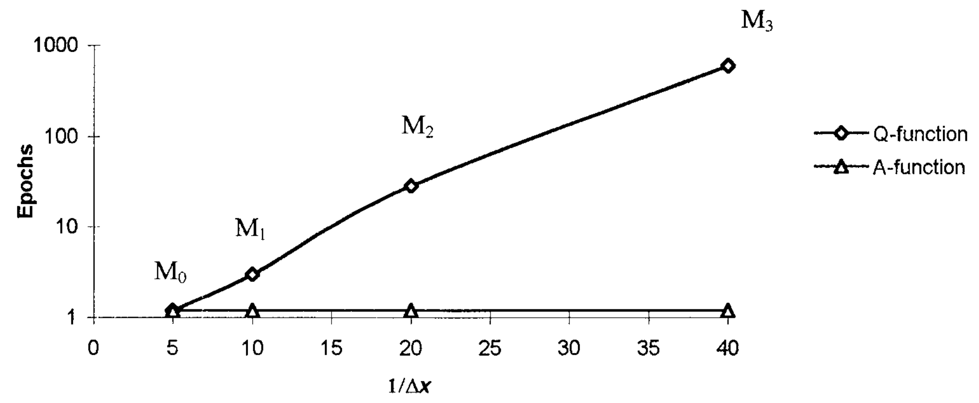


Figure 8: Comparison of time to learn optimal policy in Q-function and Advantage function

These results demonstrate that the number of epochs required to achieve π^* in every state is independent of the number of states in an optimal trajectory. It is always the case that a function c exists that causes D_{sa} to remain unchanged as d_{av1} changes. Using the heuristic described above for choosing the function c resulted in a D_{sa} of approximately 2.2 for each of the Advantage functions. Therefore, we performed supervised learning on essentially the same objective function for each MDP. This is demonstrated in Figures 9-12 below.

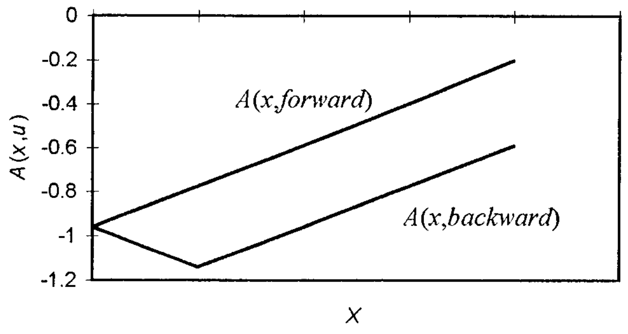


Figure 9: MDP M_0, D_{sa}=2.04

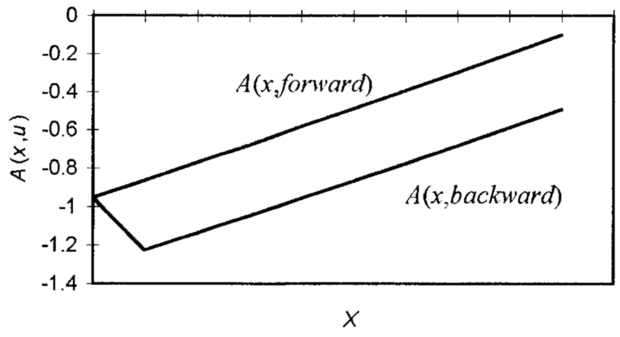


Figure 10: MDP M_1, D_{sa}=2.28

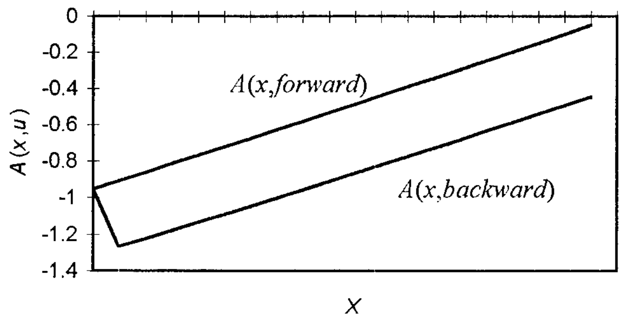


Figure 11: MDP M_2, D_{sa}=2.40

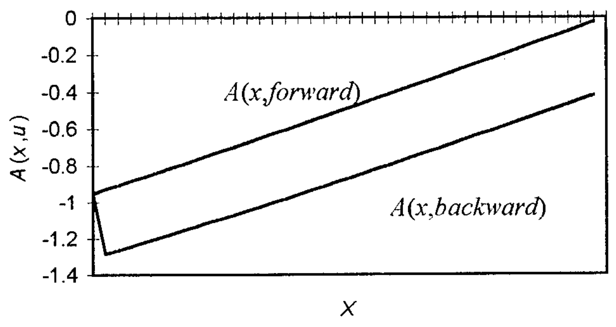


Figure 12: MDP M$_3$, D_{sa}=2.46

4.3 Semi-Markov Decision Processes: Addressing D$_{aa}$

Section 3.2.1. defined variables to describe two distinct properties of action-value functions. We illustrated the effects of the first, state-action dynamic range (D_{sa}), on the degree of sensitivity in the policy with respect to small changes in the parameter vector of the function approximator. Here we address the second property, action-action dynamic range (D_{aa}), and discuss how it may exaggerate sensitivity.

4.3.1. Action-Action Dynamic Range and d_{av2}

In Section 3.2.1. $D_{aa}(x,u)$ was defined to be equal to $d_{av1}(x) / d_{av2}(x,u)$, where $d_{av2}(x,u) = \left[\max_u Q(x,u)\right] - Q(x,u)$ and $d_{av1}(x) = \max_u Q(x,u) - \min_u Q(x,u)$. This ratio is demonstrated graphically in Figure 13.

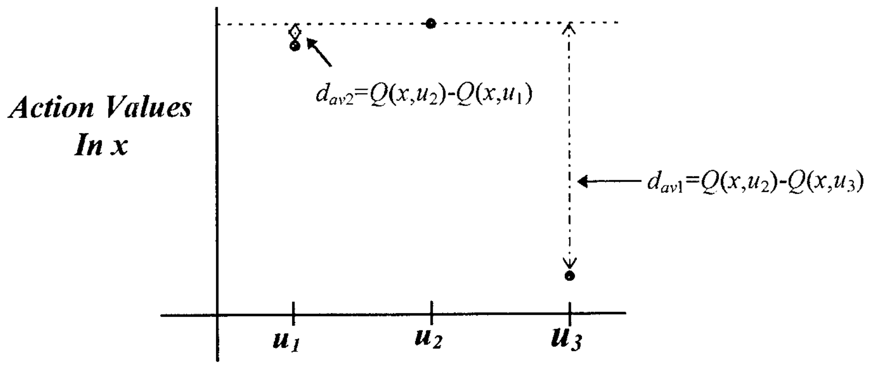


Figure 13: $D_{\mathrm{aa}}(x,u_1)=d_{av1}(x)/d_{av2}(x,u_1)$

It may be that the degree of sensitivity in the implied policy in state x is directly related to the magnitude of $\max_u D_{aa}(x,u)$. Why should this be the case?

Consider a variation of the Hall problem presented in Section 4.2. In this variation the hall is replaced with a tight rope suspended between two platforms high above a concrete floor. An RL acrobat, with perfect balance, is given the task of crossing from one platform to the other in the fewest number of steps. The class of MDPs is parameterized by the height of the platform from the floor. In each state the acrobat can perform one of three possible actions: 1) step forward, 2) step backward, and 3) step to the side. Each step moves 1 foot along the rope and incurs unit cost. A step to the side (a step off of the rope) incurs a cost equal to the height of the platform from the floor, however the acrobat is secured by a tether that allows her to climb back to the location on the tight rope from which she fell with no cost. All costs (negative rewards) are undiscounted. The length of the tight rope is 50 feet, resulting in d_{sv}=50. If we assume a platform height of 51 feet, then D_{sa}=1 for all x, the ideal state-action dynamic range. However, a new problem now exists. A plot of the values for the three actions in state 25 (the middle of the rope) is presented in Figure 14.

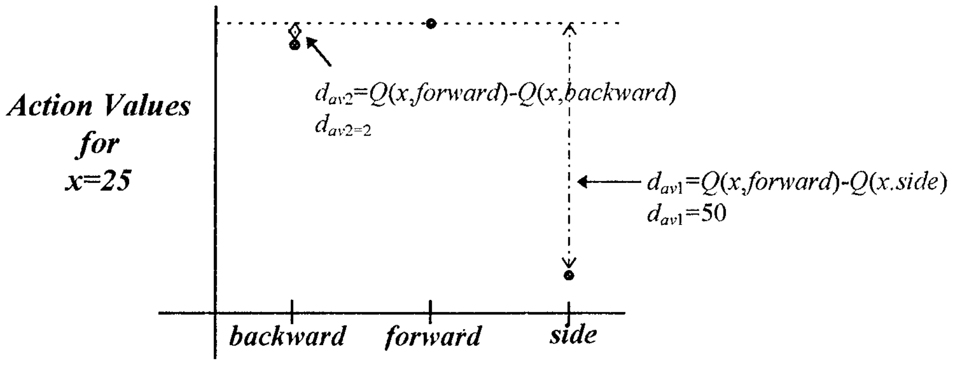


Figure 14: D_{sa}=1, D_{aa}=25

To adequately approximate π^*, the function approximator must accurately reflect the ordering of the values for *forward* and *backward*. The acrobat will devote much function approximator resources to representing the knowledge that stepping to the side will result in a very large cost, but this will come at the expense of accurately discriminating between the values of stepping forward and stepping backward. The degree of accuracy required for differentiating between these two actions increases with the height of the platform. This will necessarily result in an increase in the sensitivity in the implied policy as well. Consider what happens if we double the height of the platform. Figure 15 demonstrates that it becomes increasingly difficult to differentiate between the values of *forward* and *backward.* Essentially, the acrobat will quickly learn not to take a step to the side, but it will not easily be able to determine if it should step forward or backward. As we discuss in the next section, approximating Advantage functions rather than Q-functions solves this problem as well, given an appropriate choice of c.

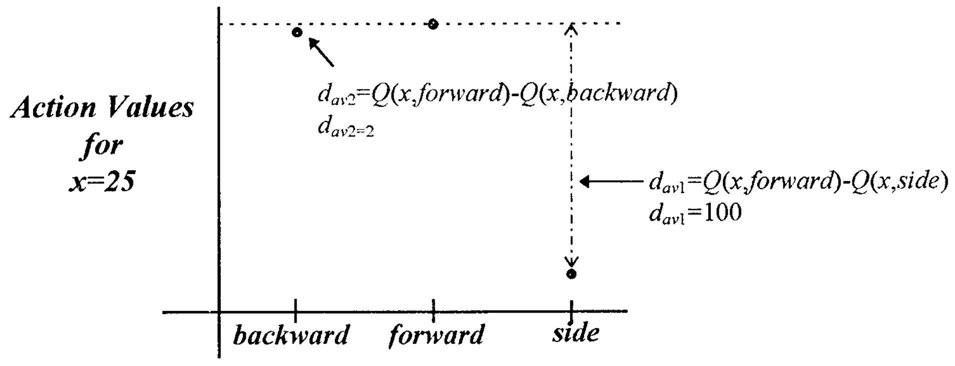


Figure 15: D_{sa}=0.5, D_{aa}=50

4.3.2. Semi-Markov Decision Processes

In previous sections we have assumed that actions are performed at each of a sequence of unit time intervals. However, if using a hierarchical RL framework, actions at any of the abstract levels in the hierarchy may be made at varying integral multiples of the unit time interval. The interval between actions may be predetermined or random. Also, if a continuous-time decision problem is treated as a discrete-time system where actions are made upon change of state, actions may be made at varying time intervals. In these cases, the framework is known as a *semi-Markov decision process* (SMDP). Such processes can be treated the same as MDPs to a large extent by taking the reward on each discrete transition as the integral of the reward over the corresponding continuous-time interval for the continuous-time case, or the sum of the rewards over the duration of the corresponding sequence of unit time intervals in the discrete-time case.

In the SMDP framework it may be likely that D_{aa} is quite large. If rewards for actions are a function of the duration of the transition, then it will be quite common for action values within a given state to differ greatly, resulting in a large D_{aa}. In the special case that the action set within a given state is a combination of "primitive" actions (actions that have a duration of unit time) and "macro" actions (actions that have a duration of more than unit time), D_{aa} will very likely be large. Only in the case that the macro actions have roughly the same values as the primitive actions will D_{aa} not be an issue with regard to sensitivity.

Large action-action dynamic range ratios are not restricted to SMDPs. They may also occur in MDPs, as demonstrated in Section 4.3.1.

4.3.3 Revised Hall Problem

To demonstrate the relationship between sensitivity and D_{aa} we again use a variation of the Hall problem. The problem specification is changed by adding macro actions to the action set in each state. The length of the hallway is fixed at one hundred primitive steps. Specifically, $X = \{-49,-48,-47,\ldots,-1,0,1,\ldots,48,49,50\}$ and $U_n = \{mleft, left, right, mright\}$ where *mleft* and *mright* are macro actions and *n* is the length of these actions measured in state transitions. The class of MDPs, M, is parameterized by *n*. State space "wraps around" forming a cycle. For example, $f(50, right) = -49$. As in the original problem specification, there exists a single terminal state, *x*=0. The reward for each primitive action is 0.01. The reward for each macro action is *n*(.01) where *n* is the number of state transitions.

In each of the examples presented below, we assume the rewards are not discounted and are being minimized. We begin by comparing the *Q*-function and Advantage function for

M_1. In M_1 the primitive actions and macro actions are of the same duration, 1 state transition.

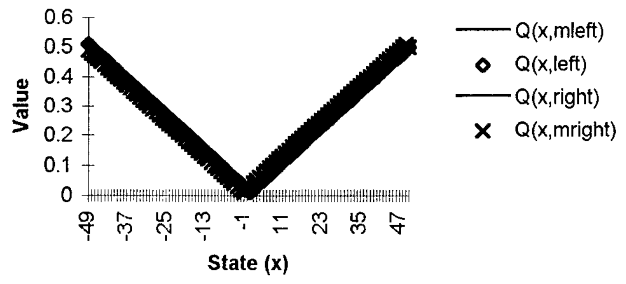


Figure 16: The optimal Q-function for M_1, the length of the macro actions is 1 unit. D_{sa}=25, D_{aa}=1.

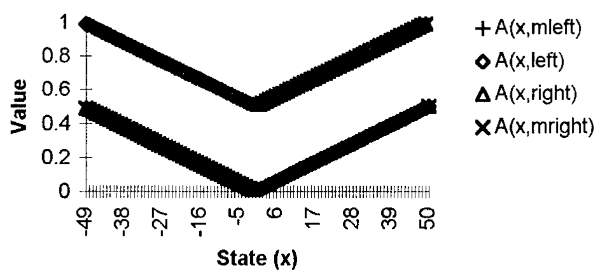


Figure 17: The optimal Advantage function for M_1 with c=25 for all state-action pairs, including macros. D_{sa}=1, D_{aa}=1

The state-action dynamic range ratio, D_{sa}, is quite large in the optimal Q-function for MDP M_1, shown in Figure 16. A choice of $c(x,u)$=25 for all state-action pairs (a choice of c^*) results in an Advantage function with a D_{sa} ratio of 1. The action-action dynamic range ratio, D_{aa}, for both functions is 1 because the duration of the macro actions equals the duration of the primitive actions. However, it is important to consider how D_{aa}

changes as the duration of the macro actions increases. If we maintain a value of 25 for all state-action pairs in c, then $D_{aa}\approx 2$ in M_2, $D_{aa}\approx 4$ in M_4, and $D_{aa}\approx 8$ in M_8.

The optimal Q-function for MDP M_{16} is presented in Figure 18 below. Each macro action has a duration of 16 state transitions. The Q-function now has 2 large dynamic range ratios, $D_{sa}\approx 25$ and $D_{aa}\approx 16$. These ratios are presented as approximations because the actual values now vary as a function of state and action.

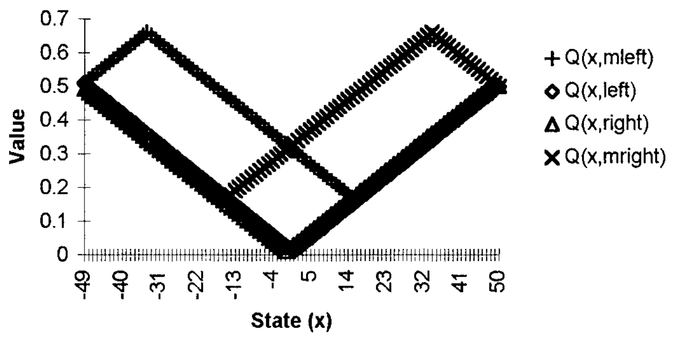


Figure 18: The optimal Q-function for MDP M_{16}. $D_{sa}\approx 25$, $D_{aa}\approx 16$.

The Advantage function for MDP M_{16}, choosing the optimal c for each state-action pair, is presented in Figure 19. Note that using c^* results in each sub-optimal action have unit distance from the optimal action in each state.

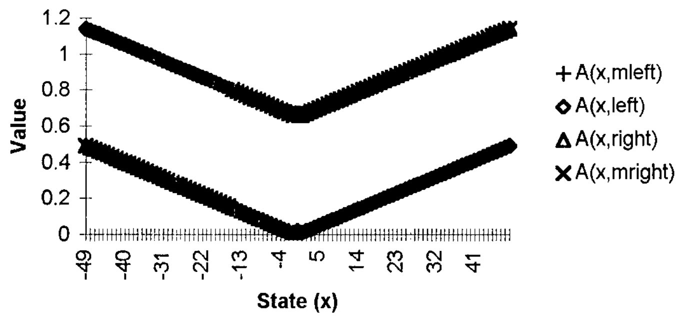


Figure 19: The optimal Advantage function for MDP M_{16} using c^*. $D_{sa}=1$, $D_{aa}=1$.

However, as stated in Section 4.1, it is rarely the case that we have enough information to determine the function c^*. So it is reasonable to question the quality of the Advantage function that results from using heuristics to choose the function c. In the Hall problem presented in Section 4.2 we chose to use a constant value for the c function. The value chosen for c was based on the change in state, Δx, resulting from performing a single action. This accomplished the goal of re-scaling the state-action dynamic range ratio, D_{sa}, and resulted in greatly decreasing the time required to achieve the stopping criteria in our experiments. Can we expect a constant scaling function c to produce similar results for SMDPs? As stated earlier, as the duration of the macro actions increases the action-action dynamic range ratio also increases. The Advantage function for MDP M_{16} with a choice of $c(x,u)$=25 for all state-action pairs is presented in Figure 20 below.

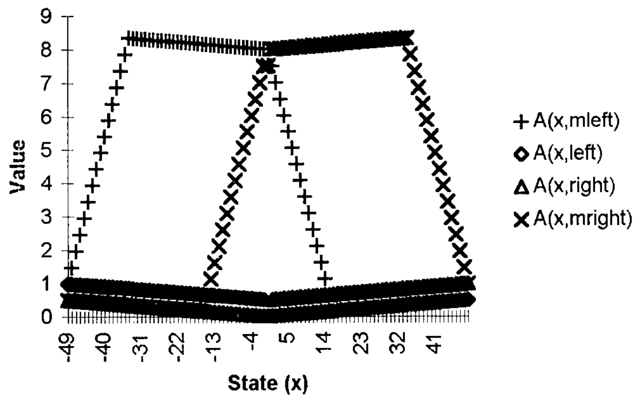


Figure 20: The optimal Advantage function for MDP M_{16} with a choice of $c(x,u)$=25 for all state-action pairs. $D_{aa}\approx 16$

This choice of c results in an action-action dynamic range ratio, D_{aa}, as large as 16 in many states. However, Figure 20 suggests a heuristic. Let the term *regret* be defined as the change in the total reward received as a result of performing a single sub-optimal action. For the problem at hand, we define a single unit of regret to be equal to the change in the total reward received as a result of performing a single sub-optimal primitive action. Using this definition it is clear that performing a single sub-optimal macro action can result in as much as 16 units of regret (16 times more than a single primitive action). This suggests scaling the value of $c(x,u)$ by a factor of 1/16 for all macro actions. More generally, let the choice of c be a function of the duration of the action. For MDP M_1, the duration of the macro actions and primitive actions are the same. Therefore we can use a constant scaling factor of 25 for all state-action pairs. For MDP M_4 let $c(x,mleft)=c(x,left)/4=6.25$, and in M_8 $c(x,mleft)=c(x,left)/8=3.125$. Using this heuristic, we construct the following function: $c(x,u)=K/l(u)$ where K is the state-action dynamic range ratio scaling factor and has a value of 25, and $l(u)$ is the action-action dynamic range ratio scaling factor and equals the duration of action u measured in state transitions. The resulting Advantage function is shown in Figure 21.

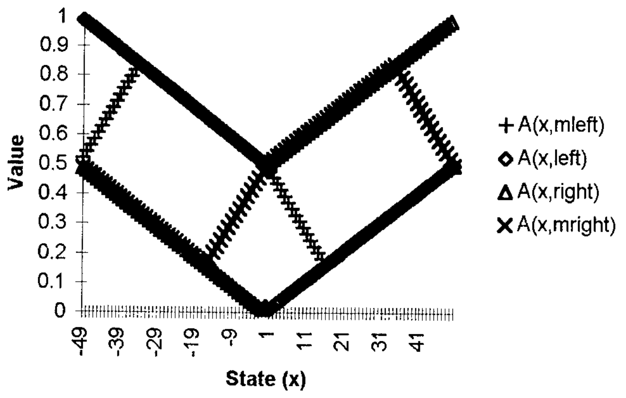


Figure 21: The Advantage function resulting from a choice of $c(x,u)=K/l(u)$ where K is the D_{sa} scaling factor and $l(u)$ is the D_{aa} scaling factor.

4.3.4 Empirical Results

Again, we performed supervised learning experiments to demonstrate the desirability of approximating Advantage functions over Q-functions. In our experiments the optimal action-value functions of MDPs M_1, M_2, M_4, and M_{16} were used as objective functions. In each experiment a double-hidden-layer, sigmoidal network was used to represent the action-value function. Each hidden layer contained 2 bipolar sigmoids with a range (-1,1). The inputs of the network were the state, x, the action, u, and a bias. All parameters were initialized to random values between -1 and 1.

Again, by a *trial* we mean the process of performing parameter updates until the stopping criteria is met. The stopping criterion is to achieve an approximation of the action-value function that implies the optimal policy in every state. Batch weight updates were performed; the parameters of the function approximator were updated only after the presentation of all input-output pairs. The performance measure was the number of parameter updates required before achieving the stopping criterion.

After much effort and optimization, the stopping criteria was never achieved when approximating the optimal *Q*-function for the simplest MDP in the class, M_1 (see Figure 16). However, the stopping criteria was achieved after 650 epochs when approximating the optimal Advantage function for the same MDP (see Figure 17).

Averaged over multiple trials with different random number seeds, the system performed an average of 650 epochs before achieving the stopping criteria for MDPs M_1, M_2, and M_4. The system performed an average of 850 epochs before achieving the stopping criteria for MDP M_{16} (see Figure 21).

To demonstrate the effects of a large action-action dynamic range ratio, D_{aa}, in a second set of experiments we chose to use a constant scaling function c with a value of 25 (c^* for MDP M_1) for MDP M_{16}. This choice of c resulted in an action-action dynamic range ratio of 16 in many of the states. As expected, even after much optimization the system was never able to achieve the stopping criteria.

5. Conclusion

We have presented one approach to reducing sensitivity resulting from a large state-action dynamic range and/or a large action-action dynamic range. One might ask if it is possible to address sensitivity issues by choosing an appropriate function approximator rather then changing the objective function. The answer is yes. Specifically, one might eliminate all sensitivity resulting from a large action-action dynamic range by using a separate function approximator for each action. Likewise, one might eliminate all sensitivity resulting from a large state-action dynamic range by using a separate function approximator for each state. Of course, this solution relegates our choice of function approximator to a simple look-up table.

It will certainly be possible in some cases to hand craft a function approximator to work with a given MDP, given enough *a priori* information about the optimal action-value function. However, we propose that approximating Advantage functions is a much more general and robust approach.

Appendix

Advantage Learning

Given Equation 3 we can derive an RL algorithm that finds an approximation of the Advantage function. We call this algorithm *Advantage Learning* (Harmon and Baird, 1996a). We begin by constructing a Bellman equation for Advantage Learning.

$$\begin{aligned}
A(x,u) &= \max_{u'} Q(x,u') - c(x,u)\left[\max_{u'} Q(x,u') - Q(x,u)\right] \\
A(x,u) &= \max_{u'} Q(x,u') - c(x,u)\left[\max_{u'} Q(x,u') - E\left(R(x,u) + \gamma \max_{u} Q(x',u)\right)\right] \\
A(x,u) &= \max_{u'} A(x,u') - c(x,u)\left[\max_{u'} A(x,u') - E\left(R(x,u) + \gamma \max_{u} A(x',u)\right)\right] \qquad (4)
\end{aligned}$$

where E indicates the expected value of performing action u in state x, and x' is the state resulting from choosing action u in state x. A standard backup operation is given in Equation 5.

$$A(x,u) \overset{\alpha}{\leftarrow} \max_{u'} A(x,u') - c(x,u)\left[\max_{u'} A(x,u') - E\left(R(x,u) + \gamma \max_{u} A(x',u)\right)\right] \qquad (5)$$

However, for reasons beyond the scope of this document, Equation 5 is not guaranteed to converge when using a lookup table as the function approximator. Therefore, we define

Advantage Learning as a *residual* algorithm (Baird, 1995). The Bellman residual is the difference in the two sides of Equation 4. The *mean squared Bellman residual* for an MDP with n states is therefore defined to be:

$$MSBR = \frac{1}{n}\sum\left(\max_{u'} A(x,u') - c(x,u)\left[\max_{u'} A(x,u') - E\Big(R(x,u) + \gamma \max_{u} A(x',u)\Big)\right] - A(x,u)\right)^2$$

Advantage Learning performs gradient descent on the *MSBR*, and is by definition a *residual* algorithm. The vector of parameters, **W**, in the function approximation system is updated according to Equation 6 below.

$$\Delta\mathbf{W} = -\alpha\left(\max_{u'} A(x,u') - c(x,u)\left[\left\langle R(x,u) + \gamma \max_{u'} A(x',u')\right\rangle - \max_{u} A(x,u)\right] - A(x,u)\right) \bullet$$
$$\left(\phi\frac{\partial}{\partial\mathbf{W}}\max_{u'} A(x,u') - \phi c(x,u)\left[\gamma\frac{\partial}{\partial\mathbf{W}}\max_{u'} A(x',u') - \frac{\partial}{\partial\mathbf{W}}\max_{u} A(x,u)\right] - \frac{\partial}{\partial\mathbf{W}}A(x,u)\right)$$

(6)

where α is the step size parameter and ϕ is a constant that controls a trade-off between pure gradient descent (when ϕ equals 1) and a fast direct algorithm (when ϕ equals 0). For a full discussion of residual algorithms see Baird(1995).

References

Albus, J. S. (1981). *Brain, Behavior, and Robotics.* Byte Books, Peterborough, NH.

Baird, L. C. (1993). *Advantage Updating.* (Technical Report WL-TR-93-1146). Wright-Patterson Air Force Base Ohio: Wright Laboratory. (available from the Defense Technical Information Center, Cameron Station, Alexandria, VA 22304-6145).

Baird, L. C. (1994). Reinforcement Learning in Continuous Time: Advantage Updating *Proceedings of the International Conference on Neural Networks.* Orlando, FL. June.

Harmon, M. E., Baird, L. C., and Klopf, A. H. (1995). Advantage Updating Applied to a Differential Game. Gerald Tesauro, et. al., eds. *Advances in Neural Information Processing Systems 7.* pp. 353-360. MIT Press, 1995.

Baird, L. C. (1995). Residual Algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 30-37. Morgan Kaufmann, San Francisco.

Harmon, M. E., Baird, L. C., and Klopf, A. H. (1996). Reinforcement Learning Applied to a Differential Game. *Adaptive Behavior*, MIT Press, (4)1, pp. 3-28.

Baird, L. C., Harmon, M. E., and Klopf, A. H. (1996). Reinforcement Learning: An Alternative Approach to Machine Intelligence. *CrossTalk, The Journal of Defense Software Engineering*, (9)2, pp. 22-24.

Harmon, M. E., and Baird, L. C. (1996a). Multi-player residual advantage learning with general function approximation. (Technical Report WL-TR-96-1065). Wright-Patterson Air Force Base Ohio: Wright Laboratory. (available from the Defense Technical Information Center, Cameron Station, Alexandria, VA 22304-6145).

Harmon, M. E., and Baird, L.C. (1996b). *Residual Advantage Learning Applied to a Differential Game.* Proceedings of the International Conference on Neural Networks (ICNN'96), Washington D.C., 3-6 June.

Puterman, M. L. (1994). *Markov Decision Processes.* New York: John Wiley & Sons, Inc.

Weaver, S., Baird, L., and Polycarpou, M. (1998). An Analytical Framework for Local Feedfoward Networks. IEEE Transactions on Neural Networks.

Watkins C. J. C. H. (1989). *Learning from Delayed Rewards.* Ph.D. thesis, Cambridge University.

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 10.Jan.00	3. REPORT TYPE AND DATES COVERED THESIS	
4. TITLE AND SUBTITLE APPROXIMATING ACTION-VALUE FUNCTIONS: ADDRESSING ISSUES OF DYNAMIC RANGE			5. FUNDING NUMBERS
6. AUTHOR(S) 1ST LT HARMON MANCE E			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) UNIVERSITY OF MASSACHUSETTS AMHERST			8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) THE DEPARTMENT OF THE AIR FORCE AFIT/CIA, BLDG 125 2950 P STREET WPAFB OH 45433			10. SPONSORING/MONITORING AGENCY REPORT NUMBER FY00-18
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION AVAILABILITY STATEMENT Unlimited distribution In Accordance With AFI 35-205/AFIT Sup 1			12b. DISTRIBUTION CODE
13. ABSTRACT *(Maximum 200 words)*			
14. SUBJECT TERMS			15. NUMBER OF PAGES 33
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT